Indian Institute of Information Technology, Allahabad

Instruction Sets of ARM, MIPS and X86 [First Draft]

CheatSheet

[Acknowledgement and Credit: Prepared by referring various Online Sources. Thanks to all authors]

Command	**ARM**			**MIPS**		**X86**	
Add	ADD D, S1,S2	ADD r1,r2,r3	r1 = r2 + r3	add $1,$2,$3	$1=$2+$3	add <reg>, <reg>	
Add						LEA EAX, [EAX + EBX + 1234567]	Perform addition with either two or three operands
ADD Carry	ADC Rh,Rh,Ra2						
Decrement						dec eax	
Increment						inc eax	
Subtract	SUB D, S1,S2	SUB r1,r2,r3	r1 = r2 – r3	sub $1,$2,$3	$1=$2-$3	sub al, ah	
add immediate				addi $1,$2,100	$1=$2+100		
add unsigned				addu $1,$2,$3	$1=$2+$3		
Multiply				mult $2,$3	$hi, $low=$2*$3 Upper 32 bits stored in special register hi Lower 32 bits stored in special register lo [Considering 32 bit architecture]	imul eax, [var]	multiply the contents of EAX by the 32-bit contents of the memory location var. Store the result in EAX.
Divide				div $2,$3	$hi,$low=$2/$3 Remainder stored in special register **hi** Quotient stored in special register**lo**	idiv eax, [var]	
Multiply (without overflow)				mul $1,$2,$3	$1=$2*$3 Result is only 32 bits!		
Load	LDR D,[offset,Constant]	LDR r1, [r2, #20]	r1= memory[r2+20]	lw $1,100($2)	$1=Memory[$2+100]; Copy from memory to register		
Store	STR Register,[offset, Constant]	STR r1, [r2, #20]	memory[r2+20] = r1	sw $1,100($2)	Memory[$2+100]=$1 Copy from register to memory		
Load Register Byte	LDRB D,[offset,Constant]	LDRB r1, [r2, #20]	r1= memory[r2+20]				
Load Immediate				li $1,100	$1=100; Loads immediate value into register		
Load register byte signed	LDRBS D,[offset,Constant]	LDRBS r1, [r2, #20]	r1= memory[r2+20]				
Store Register Byte	STRB D,[offset,Constant]	STRB r1, [r2, #20]	memory[r2+20] = r1				
Swap two values	SWP v1,v2	SWP r1,r2	r1 = r2 and r2 = r1				
Move a value	MOV D,S	MOV r1, r2	r1 = r2	move $1,$2	$1=$2	mov <reg>,<reg>	

And	AND D, S1, S2	AND r1,r2,r3	r1 = r2 & r3	and $1,$2,$3	$1=$2&$3	and eax, 0fH	
OR	ORR D, S1, S2	ORR r1,r2,r3	r1 = r2 | r3	or $1,$2,$3	$1=$2|$3		
Not	MVN D, S	MVN	r1 = ~r2			not	
AND immediate				andi $1,$2,100	$1=$2&100		
OR immediate				or $1,$2,100	$1=$2 |100		
Logical Shift Left	LSL D,S,Number of Shift	LSL r1, r2, #2	r1 = r2 << 2	sll $1,$2,10	$1=$2<<10	shl eax, 1	
Logical Shift Right	LSR D,S,Number of Shift	LSR r1, r2, #2	r1 = r2 >> 2	srl $1,$2,10	$1=$2>>10	shr eax, 1	
Compare two values	CMP V1, V2	CMP r1, r2				cmp <reg>,<reg>	
Branch (Jump)	BEQ Label EQ, LT, LE, etc.	BEQ 40	If (r1 == r2) go to Label 40	beq $1,$2,100	f($1==$2) go to PC+4+100	je <label> (jump when equal) jne <label> (jump when not equal)	
Branch (Jump)Always	B 2500	B 40	go to 2500	j 1000	go to address 1000	jmp <label>	
Branch (Jump) and Link	BL 2500	BL 2500	r14 = next statement; goto 2500; r14 is the link register				
print_int				Print integer number (32 bit)			
Read int				Read integer number from use			
Stack PUSH						push eip + sizeof(CALL); push 0xBAR [Update stack]	save return address Subtract 4 from ESP andInsert data on the stack
Stack POP						RET pop EAX	Removes data from the stack Saves in register or memory and Adds 4 to ESP
Registers	16 (32 bit registers)	R0 to r12 – General Purpose SP – Stack Pointer LR – Link Register PC – Program Counter		MIPS R2000 CPU 32 registers	0 Constant ; 1 Reserved for the assemble; 2 - 3 Result Registers; 4 - 7 Argument Registers 1 ·· · 4.; 8 - 15, 24 - 25 Temporary Registers 0 · · · 9; 16 - 23 Saved Registers 0 ·· · 7; 26 – 27 Kernel Registers 0 ·· · 1. gp - Global Data Pointer.	8 (32 bit registers)	General Purpose EAX,EBX,ECX,EDX, ESI, EDI ESP – Stack Pointer EBP – Base pointer

					sp - Stack Pointer. fp - Frame Pointer. ra - Return Address		

References

https://www.dsi.unive.it/~gasparetto/materials/MIPS_Instruction_Set.pdf [accessed on 28/4/2022]
https://www.ics.uci.edu/~aburtsev/143A/lectures/lecture03-x86-asm/lecture03-x86-asm.pdf [accessed on 28/4/2022]
https://iitd-plos.github.io/col718/ref/arm-instructionset.pdf [accessed on 28/4/2022]
https://mathcs.holycross.edu/~csci226/MIPS/summaryHO.pdf [accessed on 28/4/2022]
Computer Organization and Design (ARM edition) - The Hardware and Software Interface by David A. Patterson and John L. Hennessy